

MULTIMEDIA



UNIVERSITY

STUDENT ID NO

--	--	--	--	--	--	--	--	--	--

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2015/2016

TAI1361 – ARTIFICIAL INTELLIGENCE I

(All Sections/Groups)

6 October 2015
2:30 PM – 4:30 PM
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This question paper consists of **5** pages, excluding this cover page, with **8** questions, which are divided into **3** sections.
2. Attempt **6** out of **8** questions: **2** from **Section A**, **2** from **Section B**, and **2** from **Section C**. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the answer booklet provided.

Section A: LISP Programming**Question 1**

- (a) What does LISP stand for? (1 mark)
- (b) Give a brief description for the following statements about LISP:
- (i) LISP is bottom-up programming.
 - (ii) LISP is dynamic language.
 - (iii) LISP is functional language.
 - (iv) LISP makes no distinction between programs and data. (6 marks)
- (c) After evaluating *(setq alpha '(1 2 3) beta '(a b c) gamma '(do re mi) empty '())*, what would be the returned value of each of the following LISP expressions:
- (i) (cons alpha beta)
 - (ii) (cons (car alpha) beta)
 - (iii) (cons (car gamma) (cons (car beta) (cons (car alpha) empty))) (3 marks)

Question 2

- (a) What is a predicate in LISP? (1 mark)
- (b) What will be the returned value of the following expressions?
- (i) *(mapcar #'(lambda (x) (+ x 1)) '(1 2 3) '(4 5 6) '(-1 -2 -3))*
 - (ii) *(maplist #'append '(1 2 3) '(4 5 6) '(-1 -2 -3))* (4 marks)
- (c) Write a recursive function *count-atoms* that counts the number of atoms that appear at *all* levels of a list. For example,

(count-atoms '(people (who (lost (their houses) (in (the fire))))))

returns 8. (5 marks)

Continued.....

Question 3

(a) State whether the following statements about LISP is *TRUE* or *FALSE*:

- (i) The LISP function *eq* is a test of pointers.
- (ii) In LISP, *nil* is considered both an atom and a list.
- (iii) If *L* is bound to a non-nil list, then *(eq L (reverse (reverse L)))* always evaluates to *T*.
- (iv) The *lambda* special form can be used to define “anonymous” functions in LISP.
- (v) Every s-expression in LISP returns a value after evaluated.
- (vi) LISP does not allow you to assign a variable more than once. (3 marks)

(b) What will be the returned value of the following expression?

```
(do ((x 1 (+ x 1))
      (y 1 (+ y 2)))
    (> x 5) y)
(print y)
(print 'good))
```

(3 marks)

(c) Write a recursive function *recursive-assoc* that “looks up” a value in a list of lists, and returns the first list it finds whose first element matches a desired value. For example,

```
(recursive-assoc 'helms '((bush republican) (helms republican) (clinton
democrat)))
```

returns *(helms republican)*.

Note that this is true even if there is another entry further down the list. E.g.,

```
(recursive-assoc 'helms '((bush republican) (helms republican) (clinton democrat)
(helms conservative)))
```

still returns *(helms republican)*.

(4 marks)

Continued.....

Section B: PROLOG Programming**Question 4**

- (a) Suppose that someone has already written PROLOG predicates that define the following relationships:

<i>male(X)</i>	<i>/* X is a male */</i>
<i>female(X)</i>	<i>/* X is a female */</i>
<i>parent(X, Y)</i>	<i>/* X is a parent of Y */</i>

Write PROLOG rules to define the following relationship by using the above predicates:

daughter(X, Y) */* X is a daughter of Y */* (2 marks)

- (b) Write a recursive rule to define the following relationship by using the predicates defined in sub-section 4 (a):

grandparent(X, Y) */* X is a grandparent of Y */* (2 marks)

- (c) Given the program

<i>subset([], []).</i>	<i>% 1</i>
<i>subset(S, [H T]) :- subset(S, T).</i>	<i>% 2</i>
<i>subset([H R], [H T]) :- subset(R, T).</i>	<i>% 3</i>

draw the Selected Linear Definite clause resolution tree (SLD tree) for the query ?-
subset([a,c], [a,b,c]). (6 marks)

Continued.....

Question 5

- (a) What is negation as failure? (1 mark)
- (b) Explain with an example that negation as failure works properly for ground literals in a goal, and it fails when there is an unbound variable in a goal. (4 marks)
- (c) The following is a PROLOG program using the predicate *not*.

```

home(X) :- not(out(X)).      /* X is at home if X is not out. */
out(sue).                   /* Sue is out. */
husband(john, sue).         /* John is Sue's husband. */

```

- (i) What is PROLOG's response to the query: *?- home(sue)*.
(ii) What is PROLOG's response to the query: *?- home(john)*.
(iii) What is PROLOG's response to the query: *?- home(X)*.
(iv) Briefly explain your answer to part (iii). (5 marks)

Question 6

- (a) What is cut (!) in PROLOG? (1 mark)
- (b) Given the following PROLOG program, draw an SLD tree for the query *?- likes(sue, snoopy)*.

```

likes(sue, X) :- cat(X), !, fail.      % 1
likes(sue, X) :- toy(X).               % 2
cat(garfield).                        % 3
toy(garfield).                        % 4
toy(snoopy).                          % 5
toy(doll).                            % 6

```

(5 marks)

- (c) *subList(S,L)* is a predicate that derives all the sub lists of *L*.
len(L, N) is a predicate that determines the length of *L* and assigns it to *N*.
conc(L1,L2,L) is a predicate that concatenates (appends) lists *L1*, with *L2* to form *L*.

Assuming that the above predicates are already defined, use them to define the predicate *divide(OriginalList, L1, L2, Pos)* that divides the list *OriginalList* into *L1* and *L2* at Position *Pos*. For example:

```

?-divide([a,b,c,d,e,f], L1,L2, 4).
L1 = [a,b,c,d],
L2 = [e,f]

```

(4 marks)

Continued.....

Section C: Logical Foundation for Artificial Intelligence**Question 7**

- (a) Briefly describe what a knowledge-based agent is. (2 marks)
- (b) Briefly describe what the Modus Ponens is. (2 marks)
- (c) With the help of a truth table, argue that Modus Ponens is sound. (2 marks)
- (d) Determine whether the following propositions are valid using truth-table approach.
- (i) $(P \vee \neg P) \rightarrow Q$
- (ii) $(P \wedge \neg P) \rightarrow Q$ (4 marks)

Question 8

- (a) Express the following four statements in first-order logic, based on the given interpretation of predicate symbols:

1. All computer scientists are rational.
2. Anyone who understands a rational person is rational.
3. There is a psychotic who understands a computer scientist.
4. Therefore, some rational person is a psychotic.

$C(x)$: x is a computer scientist

$R(x)$: x is rational

$P(x)$: x is psychotic

$U(x, y)$: x understands y

(3 marks)

- (b) Convert these four statements into conjunctive normal form, with the conclusion (statement 4) negated. (4 marks)
- (c) Demonstrate, using resolution refutation that the conclusion follows validly from the premises. (3 marks)

End of Paper